The design of a Disk Resource Manager

Arie Shoshani
March 2001

A Disk Resource Manager (DRM) is a component that controls the use of a shared disk cache in a data grid. As a shared disk, it can be used by multiple users and therefore the DRM has to have some policy that determines how much of the disk cache can be used by a given user, how long to keep a file in disk once it was cached, and which files to remove when space is needed. We note that a DRM is one type of a Storage Resource manager (SRM), the other being a Tape Resource manager (TRM), and a Hierarchical Resource Manager (HRM) that is a combination of a TRM with a DRM-managed staging disk.

Note that we use in this document the term "pinning a file". This refers to the capability to request from a DRM that has a particular file to keep it for a period of time, prior to its actual transfer. For discussion of "pinning" and "pinning strategies" refer to the companion document "Replica Types and Pinning Strategies". In this document we describe the concept of a "permanent, durable, and volatile" files. Briefly, a "permanent file is a physical file that can only be created and removed by the owner of the dataset (or the data collection). A "durable" file is a physical replica of a file that can only be created and removed by the administrator of the disk cache. A "durable" file is designated to stay in cache by the disk cache administrator (i.e. not subject to dynamic removal) until he/she decides to change its status to "volatile". A "volatile" file if a physical replica of a file that is subject to removal by the DRM according to preset policies. The need for such file types is to give control to the disk administrator to enforce his/her knowledge on the expected usage of files.

There are several issues that have to be dealt with when designing the DRM functionality. These include: getting a file into the DRM-controlled disk from another source, support a "write" capability into the DRM-controlled disk by a client, how to enforce time outs, queuing requests by the DRM, registration of files in the replica catalog, space reservations, status of a file transfer request, and collecting and providing statistics on the cache availability and usage. We describe each below.

1. Getting a file into the disk from another source

At the most basic level, a DRM must be able to support a request for getting a file from another storage resource (another disk cache or a tertiary storage system) into the disk cache. This is the case that a file is known to the client because it is registered with the replica catalog. A request to DRM to get this file (what we refer to as "getRequest") requires that the DRM first check if the file in already in its disk cache. It if is not in the cache, it makes space for the file in the disk cache, usually by removing other unused files, requesting that the file at the source SRM be pinned (if such a capability exists), initiate a file transfer from the source to its disk, monitoring the progress of the transfer, and notifying the caller when the file has been fully transferred. After a file has been transferred, the DRM may register the file with the replica catalog, so that other clients

can use it. Similarly, before a shared file is removed by DRM (to make space for new files) the DRM should remove the replica entry from the replica catalog. Whether DRMs should update the replica catalog on a per file basis or a less frequent rate is a policy determined by the data grid administrator. A requested file may be found in cache because another user has previously requested it. In this case, it is wise to share the file. We will discuss later the method for sharing files while maintaining security.

2. Pinning a file

Assuming that a DRM can hold files temporarily, it is necessary to ask it to pin a file before a file transfer from it to another DRM is initiated. This is part of facilitating file sharing between clients. Of course, one can skip this step, but then we take a chance that the file may be removed in mid transfer, or may not be there when the transfer is initiated. Even if the source DRM is well behaved in that it removes its entry from the replica catalog, it is still possible that a file is not found in the source disk cache. This stems from an obvious race condition: the client may have gotten the replica reference before the entry was removed and did not start the transfer by the time the file is removed. Thus, the requested must be able to deal with "file not found" errors in any case. However, pinning a file can reduce such mishaps. Normally, a pinned file is expected to be found in the cache. If a source system has no DRM, we have no choice but to assume that the file will be there at the time of transfer. Thus, a well designed DRM can get files from sources that do not have DRMs, but it must be prepared to deal with failures, or incomplete file transfers.

3. support a "write" capability into a DRM-controlled disk by a client

Another function by a DRM is to store files that clients generate. Typically, such files are stored temporarily and then transferred to other sites, perhaps a tape system. Thus, there is no need to register such files in the replica catalog. This function is similar to getting a file as far as making space for the file, but the DRM does not have to pin a file at the source. The client, who is obviously the owner of the file, is responsible to register it with the file catalog, if he/she chooses to do that. There are two possibilities for storing a file in a the DRM-controlled disk by a client (we refer to this as a "putRequest"). One is that the DRM allocates the space for the file and the Client performs a "gridFTP put" (the "push model"). The other is that the DRM allocates the space, and performs a "gridFTP get" (the pull model). The pull model is considered safer, since it eliminates the need to deal with clients that misbehave. Our plan is to implement the pull model first.

4. Time outs

A well behaved client is expected to release a file after it is finished using it (i.e. reading it, or copying it). However, a DRM cannot rely on that, and must have a policy for removing files. It is up to the local administrator of each DRM to determine the policy for a time out of a file. There may be a different time out for files transferred into the DRM from file pinned by the DRM. A time out policy only guarantees that a file will be in cache for the time out period. Beyond that, the file may or may not be found in cache, depending on its sharing by other clients, and the need to release space. The time out enforcement is therefore per_file_per_client. But, as explained in the "Replica Types and

Pinning Strategies" document, it is possible to enforce a time out without keeping track of the clients (so called "level-1 pinning" strategy).

4. Queuing requests

In a busy system, it is possible that a file transfer request cannot be accomplished at the time it is requested. In this case, an error “system busy” will be returned to the client. The client will then try repeatedly until the request is accepted or it may give up. To avoid this situation, a DRM can help by queuing requests. Rather than return a “system busy” response, it can queue the request, and return a status with the estimated time. The client can then accept the estimated time or abort the request. This feature is not essential for a DRM, but having it can cut down on repeated requests and can be used to enforce policy and fairness of service of clients’ requests.

5. Registration of files in the replica catalog

If the DRM is managing a shared disk cache, then any file stored in that system can potentially be needed by other clients. Thus, after a file is transferred to the DRM’s disk, it should be registered in the Replica Catalog, so that other clients know about its existence. The choice to register a file is part of the DRM’s policy. Also, the frequency of registration is a local policy. The frequency of registration can be either immediate after a file was transferred, or one can have a policy to update the replica catalog periodically. Similar to registering the addition of a replica, it is necessary to update the replica catalog when a file is removed. In this case, it is wise to remove the replica entry before the file is physically removed, but as mentioned in point 2 above this does not guarantee that the file will not be requested after is removed by a client that got the reference earlier.

6. Space reservations

Space reservations can be a problematic function to provide. On the one hand, it makes sense for clients to make plans to use a certain amount of space at a certain time for a certain duration. On the other hand, such reservations can be very wasteful of space, since a DRM must keep the space unused to guarantee a reservation. It seems that for reservations to be effective, some kind of a cost must be charged to the clients, so they have an incentive to act responsibly. A responsible behavior means that the clients use the space assigned, and release it as soon as they do not need the space. Another aspect of space assignment is how to advertise or describe the available space, since it is a function of time. This information is needed for request planning. At a minimum every DRM should be able to respond to a request for space reservation for a certain time and duration either positively or suggest the earliest time that such a reservation can be made.

7. Status of a file transfer request

After a request for file caching is made, the DRM should be capable of providing the status of the request. This includes information on whether the request is still queued, whether the transfer is in progress, and how much of the file was transferred so far. In case that the request is still queued, a time estimate should be provided as to when the request will be executed.

8. Providing statistics on the cache availability and usage
This functionality is necessary in order for clients of the DRM to plan how to use the DRM. This includes statistics on past usage of the cache and the size of its request queue over time. Such statistics could be used to determine if the cache is overused and causes a bottleneck in the data grid.

Our plan is to implement DRM with increasing functionality as follows:

The "basic DRM" functionality includes:
- keeping track of files in the disk cache
- provide pinning capability (level-1)
- support permanent/volatile files
- support time out policy
- single thread serialized service
- no call back capability

Additional functionality to be added (in priority order):
- multi-threaded concurrent file transfer
- call back capability
- provide queue support
- provide status reporting
- provide write capability
- support pinning level-2
- support fair service to clients (based on local policies)
- support reservation capability